УДК 519.1

В. А. Щербакова

МОЩНОСТНАЯ ЗАДАЧА ШТЕЙНЕРА НА ОРИЕНТИРОВАННОМ ГРАДУИРОВАННОМ ГРАФЕ

1. Введение

Данная статья посвящена одной оптимизационной проблеме — мощностной задаче Штейнера на ориентированном градуированном графе.

Наша терминология в основном соответствует используемой в литературе по теории графов (см., например, [1]). Ряд новых терминов определяется ниже.

Пусть G — ориентированный граф с корнем r. Граф G называется связным по корню r, если каждая его вершина v достижима из корня r, т.е. существует путь из r в v. Граф G называется градуированным, если он связен по корню r и для каждой вершины v длина всех путей из r в v одинакова.

Мощностная задача Штейнера на ориентированном градуированном графе формулируется следующим образом.

Дано: $G = (V, E)$ — ориентированный связный по корню r градуированный граф; H — подмножество вершин графа G, называемых помеченными.

Найти $T = (V', E')$ — дерево Штейнера, т.е. подграф графа G, связный по корню r, содержащий все вершины из H и такой, что мощность $|E'|$ минимальна.

Очевидно, что искомый подграф T обязательно будет деревом. Следовательно, мощность $|V'|$ также будет минимальной.

Мощностная задача Штейнера на ориентированном градуированном графе служит моделью многих реальных проблем, возникающих в различных сферах хозяйственной деятельности. Например, при проектировании прокатных цехов в металлургическом производстве в виде ориентированного графа представлена калибровка валков прокатного стана: вершины графа соответствуют калибрам, а дуги определяют их последовательность при прокатке каждого профиля. Граф получается градуированным потому, что любая последовательность прокатки заданного профиля проходит одинаковое количество калибров. Проблема расчета

“дерева” калибровки валков при прокатке простых сортовых профилей состоит в выборе наименьшего числа калибров (т.е. вершин), обеспечивающих возможность прокатки заданного набора характерных профилей (помеченных вершин) [2]. Эта проблема решается путем поиска дерева Штейнера в соответствующем графе.

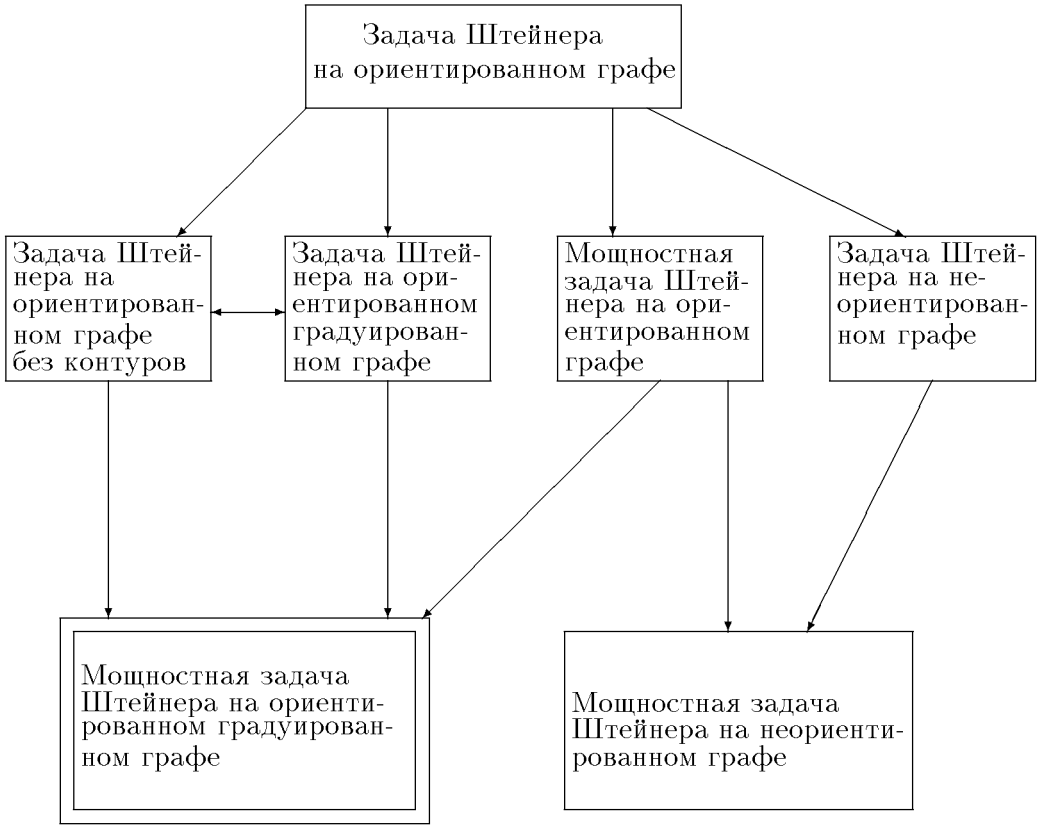

Рис.1. Схема взаимосвязей различных случаев задачи Штейнера на графах

Очевидно, что исследуемая задача является частным случаем проблемы Штейнера на ориентированном графе, которую формулируют как обобщение широко известной проблемы Штейнера на неориентированном графе. Однако общая задача Штейнера на орграфах мало исследовалась математиками. Она кратко упоминается в книге [3, с.221], формулируется в обзорах [4]–[6], посвященных, вообще говоря, задаче Штейнера на неориентированном графе. Имеются статьи [7]–[11] о задаче Штейнера

на орграфе и статья [12] для орграфа без контуров. Во всех этих работах предлагается решать задачу методом целочисленного программирования, т.е. путем минимизации целевой функции с ограничениями в виде системы линейных уравнений и неравенств с целыми коэффициентами, переменные которых принимают значения 0 или 1.

Нами был выбран подход исследования рассматриваемой задачи непосредственно на графе. Принципы такого подхода хорошо представлены в решении широко изученной задачи Штейнера на неориентированном графе. Наша задача и задача Штейнера на неориентированном графе не являются частными случаями друг друга, как это показано выше на схеме взаимосвязей различных случаев проблемы Штейнера (рис. 1).

В [13] доказана следующая

Теорема 1.1. *Мощностная задача Штейнера на ориентированном градуированном графе является NP-трудной.*

Из этой теоремы следует, что при предположении о несовпадении классов P и NP не существует полиномиального алгоритма решения исследуемой задачи.

Для решения мощностной задачи Штейнера на ориентированном градуированном графе нами был применен весь спектр методов, использующихся для исследования NP-трудных задач:

1. Алгоритмы уменьшения размера задачи.
2. Точные алгоритмы решения задачи:
   а) метод ветвей и границ,
   б) метод динамического программирования.
3. Поиск классов графов с полиномиальной разрешимостью задачи.
4. Приближенные алгоритмы.

Настоящая работа представляет собой обзор полученных при этом результатов.

2. Алгоритмы уменьшения размера задачи

2.1. Поглощение вершин и поиск доминаторов

Ниже предлагаются два простых алгоритма, позволяющих уменьшить количество непомеченных вершин и увеличить количество помеченных вершин в графе. Первый алгоритм, называемый поглощением вершин,

удаляет из графа непомеченные вершины, заведомо не лежащие в дереве Штейнера. Этот алгоритм основан на следующем утверждении [13]:

Предложение 2.1. *Пусть $T = (V', E')$ - дерево Штейнера для графа G и множества помеченных вершин H. Если для некоторой вершины v из $V' \setminus H$ найдется вершина u такая, что $In(v) \subseteq In(u)$, $Out(v) \subseteq Out(u)$ в графе G, то подграф T^*, полученный заменой в T вершины v на u, тоже является деревом Штейнера.*

Алгоритм поглощения вершин состоит из последовательного поиска и удаления из G непомеченных вершин v, для каждой из которых найдется вершина u такая, что $In(v) \subseteq In(u)$ и $Out(v) \subseteq Out(u)$. Поиск проводится до тех пор, пока такие вершины v существуют.

Сложность процедуры поиска соответствующей вершины u для одной непомеченной вершины v составляет $O(k_i \cdot (k_{i-1} + k_{i+1}))$, где k_i — количество вершин на i-м уровне графа G, содержащем v. Эта процедура выполняется для каждой непомеченной вершины и повторяется всякий раз, когда удаляется хотя бы одна вершина. Таким образом, всего процедура выполняется не больше $|V \setminus H|^2$ раз. Если удаление одной вершины выполняется за время $O(1)$, то сложность алгоритма поглощения составляет $O(S^2 \cdot k^2)$, где $S = |V \setminus H|$, $k = \max_i k_i$.

Второй алгоритм осуществляет поиск доминаторов, т.е. вершин, обязательно лежащих в дереве Штейнера. Эти вершины добавляются к множеству H помеченных вершин.

Доминатором вершины v называется вершина, лежащая на любом пути из корня r в v. Если доминатор помеченной вершины v не лежит в подграфе T графа G, содержащем все вершины из H, тогда T не будет связным по корню r, так как в T вершина v будет недостижима из r. Таким образом, доказано следующее утверждение.

Предложение 2.2. *Все доминаторы помеченной вершины v лежат в любом дереве Штейнера графа G.*

Алгоритм поиска доминаторов состоит в запуске "волны" из каждой помеченной вершины v до корня r, т.е. проводится последовательный поиск множеств: $A_1 = In(v), A_2 = In(A_1), \ldots, A_i = In(A_{i-1}), \ldots$. Если при некотором i множество A_i состоит из единственной вершины, то эта вершина и будет доминатором для v. Нетрудно видеть, что сложность этого алгоритма равна $O(h \cdot \ell \cdot k^2)$, где $h = |H|$, ℓ — число уровней вершин в графе, а k, как и выше, — наибольшее число вершин на одном уровне.

2.2. Выделение областей

Алгоритм выделения областей осуществляет разбиение исходного графа на части (области) так, что искомое дерево Штейнера может быть составлено из решений задачи Штейнера в каждой области. При этом размер любой области меньше размера исходного графа.

Алгоритм состоит из двух этапов: этапа предварительной разметки и этапа разделения графа на области.

На первом этапе выполняется поиск множества вершин W такого, что любое дерево Штейнера лежит в объединении $W \cup H$.

Обозначим через $\rho(H, v)$ наименьшее расстояние $\rho(t, v)$ по всем вершинам t из $H \setminus \{v\}$.

Определение 2.1. *Вершина v называется кустовой для вершин u и w, если u и w достижимы из v и выполняются неравенства*

$$\rho(v, u) \leq \rho(H, v), \quad \rho(v, w) \leq \rho(H, w).$$

Множество всех кустовых вершин K, которое находится с использованием следующей рекурсивной процедуры, является частью множества W. Вначале проводится поиск кустовых вершин для вершин из H, объединяемых в множество K_1. Каждая кустовая вершина для t_1 и t_2, где $t_1, t_2 \in H$, получает предварительную метку, являющуюся набором индексов, содержащим индексы t_1 и t_2. Затем проводится поиск кустовых вершин v для вершин u и w из $H \cup K_1$ при условии, что v не лежит в K_1. Если u (или w) лежит в H, то индекс u (соответственно w) приписывается к набору индексов у v; если u (или w) лежит в K_1, то к набору индексов у v приписывается весь набор индексов для u (соответственно w). Найденные кустовые вершины объединяются в множество K_2. На шаге i проводится поиск кустовых вершин v для вершин из $H \bigcup(\bigcup_{j=1}^{i-1} K_j)$ при условии, что v не лежит в $\bigcup_{j=1}^{i-1} K_j$. Поиск заканчивается, если на очередном шаге n не найдено ни одной новой кустовой вершины. Множество K получается объединением вершин из $\bigcup_{j=1}^{n-1} K_j$.

Вторая часть множества W состоит из вершин, найденных согласно правилу предварительной разметки, объединенных в множество M.

Правило предварительной разметки:

1) для каждой вершины $t \in H$ временную пометку, содержащую индекс t, получают все вершины, лежащие на путях вида $p(t^*, t)$, где $t^* \in H$ и $\rho(t^*, t) = \rho(H, t)$;

2) для каждой вершины $t \in H$, если существует последовательность кустовых вершин $\{v_1, \ldots, v_m\}$, где v_1 — кустовая вершина для t, v_i —

кустовая вершина для v_{i-1} для всех $i = 2, \ldots, m$, то временную пометку с индексом t получают все вершины, лежащие на путях вида:

а) $p(v_i, v_{i-1})$ для каждого $i = 2, \ldots, m$;

б) $p(v_1, t)$;

в) $p(t^*, v_i)$ для каждого $i = 1, \ldots, m$, где $t^* \in H$, $\rho(t^*, v_i) = \rho(H, v_i)$.

Если некоторая непомеченная вершина v получает временную пометку, содержащую индекс t (где t — вершина из H), то будем говорить, что вершины v и t удовлетворяют условию предварительной разметки. Множество W, являющееся объединением $K \cup M$, состоит из вершин, получивших временную пометку. Следующее утверждение из [13] показывает, что любое дерево Штейнера для заданных G и H будет лежать в объединении $W \cup H$.

Предложение 2.3. *Для заданных G и H в любом дереве Штейнера T для любой непомеченной вершины $v \in T$ найдется помеченная вершина t такая, что v и t удовлетворяют условию предварительной разметки.*

На втором этапе алгоритма по результатам предварительной разметки исходный граф разбивается на области. При этом дерево Штейнера для всего графа получается объединением деревьев Штейнера в каждой области.

Будем считать, что каждая помеченная вершина t имеет временную пометку, состоящую из единственного индекса t. Определим на множестве $W \cup H$ отношение τ следующим образом: выполняется $u \ \tau \ v$, если временные пометки для u и v имеют общий индекс. Пусть отношение $\hat{\tau}$ является транзитивным замыканием отношения τ. Отношение $\hat{\tau}$ будет отношением эквивалентности, следовательно, множество $W \cup H$ может быть разбито по отношению $\hat{\tau}$ на классы эквивалентности. Областью будем называть подграф, порождаемый множеством всех вершин из одного класса эквивалентности.

Алгоритм выделения областей, фактически реализующий разбиение множества $W \cup H$ на классы эквивалентности, для каждой области проводит накопление индексов, содержащихся во временных пометках вершин из этой области, и проверку существования некоторой временной пометки, пересекающейся с индексами области и имеющей новый индекс.

Сложность первого этапа выделения областей, т.е. алгоритма предварительной разметки, составляется из сложности следующих операций.

1) Выделение помеченных вершин без помеченных входов. При этом для каждой помеченной вершины просматривается ее список входящих вершин. Сложность этой операции $O(|H|\cdot k)$, где k — максимальное число вершин на одном уровне.

2) Поиск множества K кустовых вершин. Сложность этой операции $O(|H|\cdot|V\setminus H|\cdot|H|) = O(|H|^2\cdot|V\setminus H|)$ при условии, что расстояние между парой вершин находится за время $O(1)$ (т.е. все расстояния вычислены заранее).

3) Последовательный поиск кустовых вершин для кустовых вершин. При этом не более $|V\setminus H|$ раз осуществляется операция, аналогичная 2, отличающаяся от нее только тем, что t берется из множества кустов, а не из множества помеченных вершин. Так как кустовые вершины не лежат в H, получаем сложность всей операции

$$O(|V\setminus H|\cdot|V\setminus H|\cdot|H|\cdot|V\setminus H|) = O(|V\setminus H|^3\cdot|H|).$$

4) Пометка вершин, лежащих на путях между кустовыми вершинами или от помеченных вершин до кустовых вершин (см. условия предварительной разметки 2“а”,“б”,“в”). Сложность этой операции

$$O(|V\setminus H|\cdot|V|^2\cdot(|H|+|V\setminus H|)) = O(|V\setminus H|\cdot|V|^3).$$

Обозначим $|V| = n$, $|H| = h$, $|V\setminus H| = S$, а через k обозначим наибольшее число вершин на одном уровне. Тогда сложность первого этапа выделения областей составляет $O(h\cdot k+h^2\cdot S+h\cdot S^3+S\cdot n^3) = O(S\cdot n^3) \le O(n^4)$.

На втором этапе выделения областей не более h раз выполняется поиск вершин, у которых список индексов пересекается со списком индексов текущей области. Сложность алгоритма составляет $O(h\cdot n\cdot h\cdot n) = O(h^2\cdot n^2) \le O(n^4)$.

Таким образом, сложность всего алгоритма выделения областей составляет $O(S\cdot n^3 + h^2\cdot n^2) \le O(n^4)$.

3. Точные алгоритмы решения задачи

3.1. Метод ветвей и границ

Самым простым алгоритмом решения задачи является полный перебор вариантов и выбор наименьшего из них. В нашей задаче необходимо перебирать варианты, являющиеся подмножествами множества $V\setminus H$, а в качестве решения рассматривать объединение варианта с множеством H. При этом число перебираемых вариантов равно 2^S, где $S = |V\setminus H|$. Существует широко известный способ сокращенного перебора, называемый

методом ветвей и границ. Пусть Q — множество всех перебираемых вариантов. Упорядочим непомеченные вершины графа $G : \{v_1, \ldots, v_S\} = V \backslash H$. Разобьем Q на две части по следующему признаку: $Q(+v_1)$ объединяет все варианты, содержащие вершину v_1; $Q(-v_1)$ — все варианты, исключающие v_1. Каждое из множеств $Q(+v_1)$, $Q(-v_1)$ снова разобьем на две части: $Q(+v_1, +v_2)$, $Q(+v_1, -v_2)$; $Q(-v_1, +v_2)$, $Q(-v_1, -v_2)$. Продолжим разбиение до включения последней вершины v_S.

Осуществляя проход в глубину дерева поиска, можно реализовать просмотр всех вариантов и выбор наименьшего из них, удовлетворяющего условиям задачи Штейнера. Если $V^* \cup H$ порождает связный по корню r граф, то просмотр поддерева T^* не принесет решения лучшего, чем найденный вариант V^*. С другой стороны, если мощность $|V^*|$ больше либо равна мощности ранее найденного наилучшего варианта, дающего решение, то просмотр T^* снова бесполезен.

Сложность всего алгоритма равна $O((S + (S + n)) \cdot 2^S) = O(n \cdot 2^S)$.

3.2. Метод динамического программирования

Сначала определим семейство задач разной размерности, включающее решаемую индивидуальную задачу. Это семейство состоит из задач с множеством помеченных вершин $H_i \subseteq H$ и графом $G_v \subseteq G$, содержащим вершину v и все достижимые из v вершины (т.е. v — корень в G_v).

Разложимость задачи и рекуррентное соотношение, связывающее задачу большей размерности с задачами меньшей размерности, дает следующее утверждение из [14]:

Предложение 3.1. *Для любого дерева Штейнера T при заданных G и H ($|H| > 1$) существуют вершина v и разбиение множества H на непустые подмножества H_1 и H_2 такие, что $T = p(r, v) \cup T_v(H_1) \cup T_v(H_2)$, где $T_v(H_1)$, $T_v(H_2)$ — деревья Штейнера для G_v, H_1 и H_2 соответственно.*

Таким образом, получаем следующее рекуррентное соотношение:

$$|T_r(H)| = \min_{\substack{v \in V \\ (H_1, H_2)}} (\rho(r, v) + |T_v(H_1)| + |T_v(H_2)|).$$

Предлагаемый алгоритм последовательно находит $|T_v(H_i)|$, начиная с одноэлементных H_i, где $|T_v(H_i)| = \rho(v, h)$, $h = H_i$. В ходе алгоритма запоминаются тройки (v, H_1^*, H_2^*), в которых достигается минимум, используемые впоследствии для построения дерева $T_r(H)$.

Сложность алгоритма определяется следующим образом. На первом шаге на построение расстояний между всеми парами вершин требуется

$O(n^3)$ времени [15, с.132]. На шаге $i > 1$ находим $n \cdot \binom{|H|}{i}$ величин $q_v(H^*)$, для каждой из которых ищем минимум из $n \cdot \dfrac{2^i - 2}{2} = n \cdot (2^{i-1} - 1)$ элементов, где $\dfrac{2^i - 2}{2}$ – число различных разбиений H^* на два непустых подмножества. Таким образом, на шаг $i > 1$ затрачивается $n \cdot \binom{|H|}{i} \cdot n \cdot (2^{i-1} - 1)$ времени. Тогда общее время равно

$$n^3 + \sum_{i=2}^{|H|} (n^2 \cdot \binom{|H|}{i} \cdot (2^{i-1} - 1)) = n^3 + \frac{n^2}{2} \sum_{i=2}^{|H|} (\binom{|H|}{i} \cdot (2^i - 2)) \le n^3 + \frac{n^2}{2} \cdot 3^{|H|} =$$

$$= O(n^3 + n^2 \cdot 3^{|H|}).$$

Оценка для сложности по объему памяти складывается из того, что нужно хранить матрицу кратчайших расстояний и матрицу оценок деревьев Штейнера по всем подмножествам H. Таким образом, она составляет $O(n^2 + n \cdot 2^{|H|})$.

Оценка сложности по времени $O(n^3 + n^2 \cdot 3^{|H|})$ показывает, что алгоритм динамического программирования эффективен, когда помеченных вершин мало. Алгоритм ветвей и границ, сложность которого по времени составляет $O(n \cdot 2^{|V \setminus H|})$, напротив, эффективен, когда помеченных вершин много (т.е. мало непомеченных вершин). Следовательно, эти два алгоритма, дающие точное решение задачи, удачно дополняют друг друга: в зависимости от количества помеченных вершин эффективен либо один, либо другой алгоритм.

4. Полиномиально разрешимые классы задач

4.1. Задачи на ориентированных деревьях и класс $\mathbf{K}(\mathbf{TP}_2)$

Для задачи Штейнера на ориентированном графе имеется лишь предположение о полиномиальной разрешимости проблемы в случае ориентированных последовательно-параллельных графов (т.е. графов, которые при отбрасывании ориентации не содержат подграфов, гомеоморфных K_4).

Для задачи Штейнера на ориентированном градуированном графе полиномиально разрешимыми будут случаи задачи на ориентированных деревьях. Действительно, если ориентированный градуированный граф G является деревом (т.е. при отбрасывании ориентации граф G не имеет циклов), то каждая вершина в G, кроме корня r, имеет только одну

входящую вершину. Тогда для любой вершины v (кроме корня), принадлежащей дереву Штейнера, входящая в нее вершина также будет лежать в дереве Штейнера. Алгоритм поиска дерева Штейнера в ориентированном дереве последовательно строит множество вершин $V(T)$, помещая в него сначала все помеченные вершины, и затем для каждой v из $V(T)$ ищет входящую вершину и помещает ее в $V(T)$.

Очевидно, этот алгоритм имеет сложность $O(|V|)$ при условии, что хранится список $In(v)$ для каждой вершины v.

Алгоритм распознавания принадлежности произвольного ориентированного градуированного графа классу деревьев состоит в подсчете количества входящих вершин для всех v, кроме корня r. Если у всех вершин по одной входящей вершине, то G является деревом, в противном случае — нет. Этот алгоритм также линеен.

Ситуация на ориентированном дереве будет использоваться при решении задачи из классов $\mathbf{K}(\mathbf{TP}_2)$ и $\mathbf{K}(\mathbf{TS}_2)$. Задачи из этих классов определены соответственно на классах графов $\mathbf{TP}_2$ или $\mathbf{TS}_2$ и имеют однотипные ограничения на расположение помеченных вершин.

Дадим индуктивное определение класса графов $\mathbf{TP}_2$, являющегося ограничением на вид графа G.

Определение 4.1. *Назовем классом* $\mathbf{TP}_2$ *наименьший класс ориентированных графов, содержащий в себе все ориентированные деревья с корнем* r*, удовлетворяющий свойству: для любого графа* G *из* $\mathbf{TP}_2$ *граф* $G^* = (V \cup \{v\}, E \cup \{(u,v),(v,w)\})$*, где* u *и* w *лежат в* G*,* v *— новая вершина, не принадлежащая* G*, и* $\rho(u,w) = 2$ *в графе* G*, также принадлежит классу* $\mathbf{TP}_2$.

Класс $\mathbf{TP}_2$ является подклассом класса ориентированных градуированных графов, так как любое ориентированное дерево является градуированным графом, а добавление путей сохраняет градуированность.

Вершина v в графе G называется проходной, если ее полустепени захода и исхода равны 1, и существует вершина v' такая, что $In(v') \supseteq In(v)$, $Out(v') \supseteq Out(v)$. В [16] доказана

Лемма 4.1. *Если граф* G *принадлежит классу* $\mathbf{TP}_2$*, то, удаляя последовательно проходные вершины, пока такие существуют, получим ориентированное дерево с одним корнем.*

Возможность разобрать граф G из класса $\mathbf{TP}_2$ до дерева путем удаления только непомеченных вершин и является ограничением на расположение помеченных вершин в определении класса $\mathbf{K}(\mathbf{TP}_2)$.

Определение 4.2. *Граф G и множество помеченных вершин H в нем задают мощностную задачу Штейнера из класса $\mathbf{K}(\mathbf{TP}_2)$, если G принадлежит $\mathbf{TP}_2$ и его можно разобрать до дерева последовательным удалением непомеченных проходных вершин.*

Таким образом, алгоритм решения задачи из класса $\mathbf{K}(\mathbf{TP}_2)$ состоит в последовательном поиске и удалении непомеченных проходных вершин и в поиске дерева Штейнера на полученном дереве.

При условии, что хранятся $In(v)$ и $Out(v)$ для каждой вершины v графа G, а удаление вершины требует $O(1)$ времени, сложность всего алгоритма составляет $O(|V| \cdot |V| \cdot k \cdot k + |V|) = O(|V|^2 \cdot k^2) \leq O(n^4)$, где k — максимальное число вершин на одном уровне.

Алгоритм распознавания принадлежности произвольного случая задачи Штейнера классу $\mathbf{K}(\mathbf{TP}_2)$ состоит в последовательном поиске и удалении непомеченных проходных вершин и проверке принадлежности полученного графа классу ориентированных деревьев. Очевидно, что сложность этого алгоритма также составляет $O(|V|^2 \cdot k^2) \leq O(n^4)$.

Вышеприведенные определения дают конструктивное описание классов $\mathbf{TP}_2$ и $\mathbf{K}(\mathbf{TP}_2)$. В приводимых ниже теоремах дается описание классов $\mathbf{TP}_2$ и $\mathbf{K}(\mathbf{TP}_2)$ в терминах запрещенных-разрешенных подграфов и конфигураций.

В формулировках теорем используются подграфы особого вида.

Определение 4.3. *Обручем длины n назовем градуированный граф из n вершин, составленный из двух путей $p_1(u, v)$ и $p_2(u, v)$, не пересекающихся по внутренним вершинам и таких, что не существует путей между вершинами путей p_1 и p_2, кроме, быть может, пути из u в v.*

Определение 4.4. *Короной длины n назовем граф $G = (V, E)$, где $V = \{v_0, \ldots, v_{n-1}, u_0, \ldots, u_{n-1}\}$, $E = \{(v_i, u_i), (v_i, u_{i+1(\mathrm{mod} n)}) | i = 0, \ldots, n-1\}$.*

Определение 4.5. *M-конфигурацией в графе будем называть пять вершин $\{v_1, v_2, u_1, u_2, u_3\}$, между которыми существуют только следующие пути: $p(v_1, u_1)$, $p(v_1, u_2)$, $p(v_2, u_2)$, $p(v_2, u_3)$. Вершины u_1, u_3 назовем крайними в M-конфигурации, а вершину u_2 — средней.*

Определение 4.6. *N-конфигурацией в графе назовем четыре вершины $\{v_1, v_2, u_1, u_2\}$, между которыми существуют только следующие пути: $p(v_1, u_1)$, $p(v_1, u_2)$, (v_2, u_2). Вершины u_1, v_2 будем называть крайними, вершину u_1 — нижней, вершину v_2 — верхней.*

В [16] доказана следующая

Теорема 4.1. *Граф G принадлежит классу $\mathbf{TP}_2$ тогда и только тогда, когда G удовлетворяет следующим условиям:*

1) *граф G является ориентированным, градуированным, связным по корню r;*

2) *в графе G нет подграфов-обручей длины больше 4;*

3) *в графе G нет подграфов-корон длины больше 3;*

4) *в графе G нет M-конфигураций с висячими в G крайними вершинами.*

В следующей теореме, публикуемой здесь впервые, дается описание класса $\mathbf{K}(\mathbf{TP}_2)$ в терминах запретов на расположение помеченных вершин.

Теорема 4.2. *Граф G и множество H помеченных вершин в нем задают мощностную задачу Штейнера из класса $\mathbf{K}(\mathbf{TP}_2)$ тогда и только тогда, когда выполняются следующие условия:*

1) *граф G принадлежит классу $\mathbf{TP}_2$;*

2) *для каждой вершины v графа G в множестве входящих в нее вершин $In(v)$ содержится не более одной помеченной вершины;*

3) *в графе G нет N-конфигурации, в которой крайние вершины являются помеченными или листьями.*

4.2. Класс $\mathbf{K}(\mathbf{TS}_2)$

Задачи из класса $\mathbf{K}(\mathbf{TS}_2)$ имеют ограничения на вид графа G и на расположение помеченных вершин.

Дадим индуктивное определение класса графов $\mathbf{TS}_2$, являющегося ограничением на вид графа G.

Определение 4.7. *Назовем классом $\mathbf{TS}_2$ наименьший класс ориентированных графов, содержащий в себе все ориентированные деревья с корнем r и удовлетворяющий свойству: для любого графа G из $\mathbf{TS}_2$ граф $G^* = (V \cup \{v\}, E \cup \{(u,v)|u \in U\} \cup \{(v,w)|w \in W\})$, где v - новая вершина, не принадлежащая G, $U \subseteq In(v')$, $U \neq \emptyset$, $W \subseteq Out(v')$, $W \neq \emptyset$ для некоторой вершины v' из G, также принадлежит классу $\mathbf{TS}_2$.*

Очевидно, что класс $\mathbf{TS}_2$ является подклассом класса ориентированных градуированных графов и содержит в себе класс $\mathbf{TP}_2$.

Вершина v в графе G называется несущественной, если $|In(v)| > 0$, $|Out(v)| > 0$ и существует вершина v' такая, что $In(v') \supseteq In(v)$, $Out(v') \supseteq Out(v)$.

В [16] доказана следующая

Лемма 4.2. *Если граф G принадлежит классу $\mathbf{TS}_2$, то, удаляя последовательно несущественные вершины, пока такие имеются, получим ориентированное дерево с одним корнем.*

Возможность разобрать граф G из класса $\mathbf{TS}_2$ до дерева путем удаления только непомеченных вершин и является ограничением на расположение помеченных вершин в определении класса $\mathbf{K}(\mathbf{TS}_2)$.

Определение 4.8. *Граф G и множество помеченных вершин H в нем задают мощностную задачу Штейнера из класса $\mathbf{K}(\mathbf{TS_2})$, если G принадлежит $\mathbf{TS}_2$, и граф G можно разобрать до дерева последовательным удалением непомеченных несущественных вершин.*

Таким образом, алгоритм решения задачи из класса $\mathbf{K}(\mathbf{TS}_2)$ состоит в последовательном поиске и удалении непомеченных несущественных вершин и в поиске дерева Штейнера на полученном дереве.

Сложность всего алгоритма равна $O(|V|\cdot|V|\cdot k\cdot k+|V|) = O(|V|^2\cdot k^2) \leq O(n^4)$, где k — максимальное число вершин на одном уровне.

Алгоритм распознавания принадлежности произвольного случая задачи Штейнера классу $\mathbf{K}(\mathbf{TS}_2)$ состоит в последовательном поиске и удалении непомеченных несущественных вершин и проверке принадлежности полученного графа классу ориентированных деревьев. Очевидно, что сложность этого алгоритма также составляет $O(|V|^2 \cdot k^2) \leq O(n^4)$.

Вышеприведенные определения дают конструктивное описание классов $\mathbf{TS}_2$ и $\mathbf{K}(\mathbf{TS}_2)$. В приводимых ниже теоремах дается описание классов $\mathbf{TS}_2$ и $\mathbf{K}(\mathbf{TS}_2)$ в терминах запрещенных-разрешенных подграфов и конфигураций.

В [16] доказана следующая

Теорема 4.3. *Граф G принадлежит классу $\mathbf{TS}_2$ тогда и только тогда, когда G удовлетворяет следующим условиям:*

1) *граф G является ориентированным, градуированным, связным по корню r;*

2) *в графе G нет подграфов-обручей длины больше 4;*

3) *если в G есть корона длины n, то имеется общая заходящая вершина для всех нижних вершин короны;*

4) *если в G есть M-конфигурация с висячими крайними вершинами, то существует вершина w, из которой достижимы средняя и обе крайние вершины M-конфигурации и такая, что $\rho(r,w) \geq \min(\rho(r,v_1),\rho(r,v_2))$.*

Следующая теорема публикуется впервые.

Теорема 4.4. *Граф G и множество H помеченных вершин в нем задают мощностную задачу Штейнера из класса $\mathbf{K}(\mathbf{TS_2})$ тогда и только тогда, когда выполняются следующие условия:*

1) *граф G принадлежит классу $\mathbf{TS}_2$;*

2) *для каждой вершины v графа G в множестве входящих в нее вершин содержится не более одной помеченной вершины;*

3) *в графе G нет N-конфигурации, в которой крайние вершины являются помеченными или листьями.*

5. Приближенные алгоритмы решения задачи

5.1. Алгоритмы с неограниченной погрешностью

Для мощностной задачи Штейнера на ориентированном градуированном графе приближенный алгоритм строит связное по корню дерево, содержащее все помеченные вершины, количество ребер которого, возможно, не минимально.

Здесь погрешностью алгоритма будем называть отношение $\dfrac{APP}{OPT}$, где APP — число ребер в приближенном решении, а OPT — число ребер в оптимальном решении.

В этом разделе мы рассмотрим приближенные алгоритмы, использующие некоторые естественные стратегии поиска решения, и покажем, что они дают неограниченную погрешность, т.е. для любого числа α найдутся граф G и множество помеченных вершин H такие, что погрешность $\dfrac{APP}{OPT}$ больше α. Кроме того, некоторые из этих алгоритмов не являются полиномиальными.

Очевидно, что, запустив не до конца алгоритм ветвей и границ, мы можем взять в качестве приближенного наилучшее текущее решение, имеющееся в момент остановки алгоритма. Время работы такого приближенного алгоритма в наихудшем случае такое же, как у алгоритма ветвей и границ, т.е. оно зависит экспоненциально от размера задачи. Число ребер в приближенном решении в наихудшем случае равно $(|V| - 1)$. Следовательно, это — алгоритм с неограниченной погрешностью.

Идея другого алгоритма состоит в том, чтобы последовательно, начиная с последнего уровня графа, выбирать в соседнем сверху уровне наименьшее число непомеченных вершин, покрывающих все помеченные вершины текущего уровня; выбранные вершины добавляются в H и считаются в дальнейшем помеченными. Иными словами, последовательно решается мощностная задача Штейнера для орграфа с тремя уровнями вершин. Назовем такой приближенный алгоритм алгоритмом минимальных соседних покрытий. Так как в случае графа на трех уровнях задача остается NP-трудной, сложность этого алгоритма зависит экспоненциально от k, где k — максимальное число вершин на одном уровне.

Обозначим через APP_{cov} число ребер в приближенном решении, полученном алгоритмом минимальных соседних покрытий. Существует верхняя оценка погрешности этого алгоритма, зависящая от размера задачи, так как справедливо неравенство

$$\frac{APP_{cov}}{OPT} \leq \frac{|H| \cdot \ell}{|H| + \ell} \leq \frac{\max(|H|, \ell)}{2},$$

где ℓ – число уровней вершин графа G.

Следующее предложение показывает, что алгоритм минимальных соседних покрытий является алгоритмом с неограниченной погрешностью.

Предложение 5.1. *Для любого числа $\alpha > 1$ найдутся граф G и множество помеченных вершин H в нем, для которых $\dfrac{APP_{cov}}{OPT} > \alpha$.*

Доказательство. Рассмотрим граф, изображенный на рис. 2. В этом графе имеется k помеченных вершин на последнем уровне и помеченная вершина a. В графе существуют: пути $p(a, i)$ длины 2 для всех $i = 1, \ldots, k$; пути $p(r, i)$, не проходящие через a, $i = 1, \ldots, k$; вершина b, из которой идут ребра в вершины 1 и 2. Пусть $\rho(r, a) = m$.

Алгоритм минимальных соседних покрытий выберет вершину b и, возможно, вершины $c_3, \ldots, c_k$, не лежащие на путях из a в i, где $i = 3, \ldots, k$. Следовательно, $APP_{cov} = m + m + 1 + 2 + (k - 2)(m + 2) = k \cdot m + 2k - 1$. С другой стороны, очевидно, что $OPT = m + 2k$.

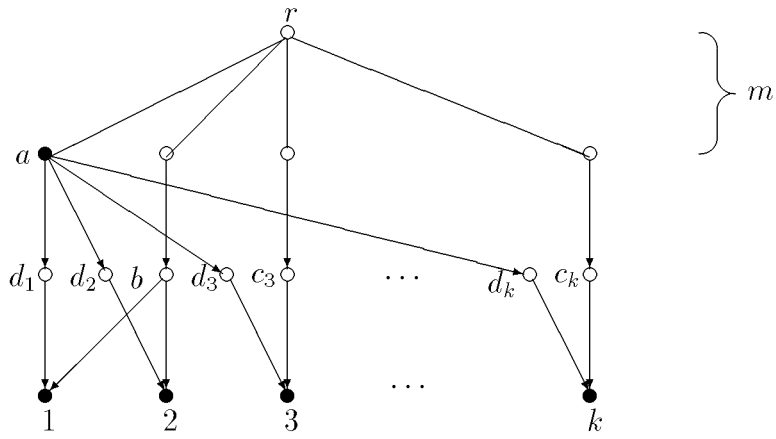

Рис.2. Граф, для которого $\frac{APP_{cov}}{OPT} > \alpha$

Пусть $k = 3\alpha$, $m = k$. Тогда выполняется

$$\frac{APP_{cov}}{OPT} = \frac{(3\alpha)^2 + 6\alpha - 1}{9\alpha} = \alpha + \frac{2}{3} - \frac{1}{9\alpha} > \alpha + \frac{5}{9} > \alpha.$$

Предложение доказано.

Идея еще одного алгоритма, называемого алгоритмом минимальных целевых путей, состоит в том, чтобы для каждой помеченной вершины v, начиная с последнего уровня, выбирать и объявлять помеченными вершины в $p(H, v)$, т.е. каждая помеченная вершина кратчайшим путем соединяется с другой помеченной вершиной, лежащей выше (или с корнем). Тогда в дереве, составленном из таких путей, каждая вершина будет достижима из корня и все помеченные вершины будут лежать в этом дереве.

Этот алгоритм полиномиален. Существует оценка погрешности

$$\frac{APP_{path}}{OPT} \leq \frac{\max(|H|, \ell)}{2},$$

где APP_{path} — число ребер в решении, полученном алгоритмом минимальных целевых путей. Этот алгоритм также является алгоритмом с неограниченной погрешностью.

Предложение 5.2. *Для любого числа $\alpha > 1$ найдутся граф G и множество помеченных вершин H в нем, для которых $\frac{APP_{path}}{OPT} > \alpha$.*

Доказательство. Рассмотрим граф на рис. 3.

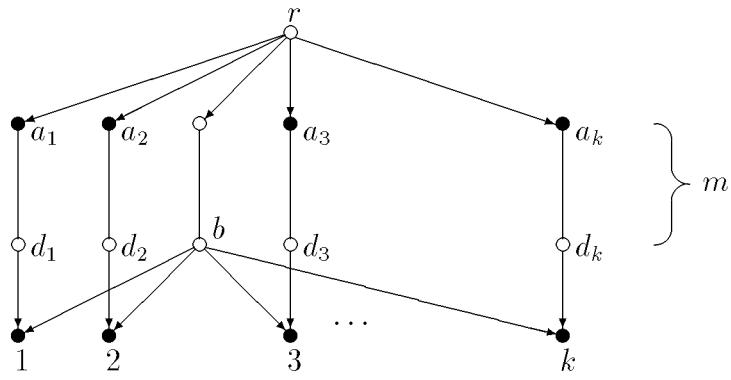

Рис.3. Граф, для которого $\frac{APP_{path}}{OPT} > \alpha$

В этом графе k помеченных вершин на последнем уровне и k помеченных вершин $\{a_1, \ldots, a_k\}$ на уровне 1, где $k > 1$. Для каждого $i = 1, \ldots, k$ существуют непересекающиеся пути $p(r, i)$, проходящие через a_i; существует вершина b, из которой идут ребра во все $i = 1, \ldots, k$; путь $p(r, b)$ не пересекается с вышеперечисленными путями.

Пусть $\rho(a_i, i) = m+1$, $i = 1, \ldots, k$. Алгоритмом минимальных целевых путей будут помечены вершины в путях $p(a_i, i)$ для каждого $i = 1, \ldots, k$. Следовательно, $APP_{path} = k + k(m+1) = k \cdot m + 2k$.

Очевидно, что $OPT = 2k + m + 1$. Пусть $m = 3\alpha$, $k = m$. Тогда

$$\frac{APP_{path}}{OPT} = \frac{9\alpha^2 + 6\alpha}{9\alpha + 1} = \alpha + \frac{5\alpha}{9\alpha + 1} > \alpha.$$

Предложение доказано.

5.2. Алгоритм с улучшенной оценкой погрешности

Предлагаемый в этом подразделе алгоритм имеет полиномиальную сложность и хорошую оценку погрешности на всех примерах общего вида, рассмотренных в предыдущем подразделе. Для произвольного графа такой хорошей оценки погрешности не найдено. Однако не найдено и ни одного семейства графов, на котором предлагаемый алгоритм давал бы неограниченную погрешность.

В алгоритме используется функция $f(v)$, определенная на вершинах

графа. Значение функции определяется по формуле

$$f(v) = \frac{\rho(H, v) + \sum_{h_i \in N_v} \rho(v, h_i)}{|N_v|},$$

где N — множество всех помеченных вершин, не имеющих помеченных входящих вершин; N_v — множество вершин из N, достижимых из v, для которых пути из v в N_v не содержат промежуточных помеченных вершин.

Вначале строится семейство $\mathcal{T} = \{T_1, \ldots, T_{|N|}\}$ одноэлементных деревьев, т.е. $T_i = \{h_i\}$, $h_i \in N$, $i = 1, \ldots, |N|$. На каждом шаге алгоритма выбирается вершина v, для которой значение $f(v)$ минимально. Функция $f(v)$ оценивает в некотором смысле "среднее расстояние" от вершины v до помеченных вершин, поэтому алгоритм будем называть алгоритмом минимального среднего расстояния. В ходе алгоритма несколько деревьев $T_{i_1}, \ldots, T_{i_{|N_v|}}$ заменяются новым деревом T_j, полученным добавлением к этим деревьям путей $p(v, r_{i_1}), \ldots, p(v, r_{i_{|N_v|}})$, где r_{i_m} — корень дерева T_{i_m}, $m = 1, \ldots, |N_v|$, v — корень нового дерева T_j. В множество H добавляются вершины этих деревьев, а множество $N = \{r_{i_m}\}$ составляется из корней деревьев в семействе $\mathcal{T}$.

Алгоритм завершается, когда в семействе $\mathcal{T}$ остается единственное дерево T_1. Если корень r_1 этого дерева совпадает с корнем r графа G, то T_1 является приближенным решением задачи, в противном случае решением является дерево $T_{mean} = T_1 \cup p(r, r_1)$.

Сложность алгоритма минимального среднего расстояния составляет $O(|V|^5)$ при условии, что расстояния вычисляются по ходу алгоритма. Если матрицу расстояний найти один раз и хранить в памяти, то сложность алгоритма будет $O(|V|^2 + |V|^3) = O(|V|^3)$.

Найдем погрешность алгоритма минимального среднего расстояния на графах, использовавшихся в предыдущем подразделе.

Рассмотрим граф, изображенный на рис. 2. Если $m \geq 2$, то для любого $i = 1, \ldots, k$ значение $f(d_i) = 2$ будет наименьшим среди всех вершин. Тогда число ребер в решении, полученном алгоритмом минимального среднего расстояния, равно $APP_{mean} = OPT = m + 2k$.

Если $m = 1$, то выполняется $f(b) = f(d_i)$, $i = 1, \ldots, k$. Тогда $APP_{mean} = 4 + (k-1) \cdot 2 + m = 3 + 2k$. Следовательно,

$$\frac{APP_{mean}}{OPT} = \frac{3 + 2k}{1 + 2k} = 1 + \frac{2}{1 + 2k}.$$

Таким образом, для любого графа G и множества H, имеющих вид как на рис. 2, выполняется $\dfrac{APP_{mean}}{OPT} < 2$.

Рассмотрим граф, изображенный на рис. 3. Если $m+1 < 1+\dfrac{m+1}{k}$, то получаем $k \cdot m < m+1$, $k < 1+\dfrac{1}{m}$, $k = 1$, т.е. это — вырожденный случай.

Если $m+1 > 1+\dfrac{m+1}{k}$, то выполняется $k > 1+\dfrac{1}{m}$. Это верно для любых $m > 1$, $k > 2$. Тогда значение $f(b) = 1+\dfrac{m+1}{k}$ будет наименьшим среди всех вершин. Получаем равенство $APP_{mean} = k+m+1+k = OPT$.

Если $m = 1$, $k = 2$, то имеем $f(b) = 2 = f(d_i), i = 1,\ldots,k$. Тогда $APP_{mean} = 4+2 = 6 = OPT$. Таким образом, для графа G и множества H таких, как на рис. 3, всегда выполняется $\dfrac{APP_{mean}}{OPT} = 1$.

До сих пор не найдены примеры графов, для которых погрешность $\dfrac{APP_{mean}}{OPT}$ была бы больше 2.

Литература

1. Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И. Лекции по теории графов. М.: Наука, 1990.
2. Гусев А.А. Разработка методов технологических расчетов при проектировании с применением ЭВМ сортопрокатных станов: Дис. ... канд. техн. наук. Свердловск, 1987.
3. Михалевич В.С., Трубин В.А., Шор Н.З. Оптимизационные задачи производственно-транспортного планирования: модели, методы, алгоритмы. М.: Наука, 1986.
4. Гордеев Э.Н., Тарасцов О.Г. Задача Штейнера: Обзор // Дискрет. математика. 1993. №2. С.3–28.
5. Winter P. Steiner problem in networks: a survey // Networks. 1987. Vol.17. P.129–167.
6. Maculan N. The Steiner problem in graphs // Annals of Discrete Math. 1987. Vol.31. P.185–212.
7. Duin C.W., Volgenant A. Some generalization of the Steiner problem in graphs // Networks. 1987. Vol.17, №3. P.353–364.
8. Liu W. A lower bound for the Steiner tree problem in directed graphs // Networks. 1990. Vol.20, №6. P.765–778.
9. Maculan N., Arpin D., Nguyen S. Le problem de Steiner sur un graphs orienté: formulations et relaxations // Matemática Aplicada e Computacional. 1988. Vol.7, №2. P.109–118.

10. Maculan N., Souza P., Candia V.A. An approach for the Steiner problem in directed graphs // Annals Operations Research. 1991. Vol.33. P.471–480.

11. Wong R.T. A dual ascent approach for Steiner tree problems on a directed graph // Math. Programming. 1984. Vol.28. P.271–287.

12. Nastansky L., Selkow S.M., Stewart N.F. Cost-minimal trees in directed acyclic graphs //Ztschr. für Operations Research. 1974. Vol.18. P.59–67.

13. Щербакова В.А. Минимальные деревья, связывающие заданное подмножество вершин в ориентированном градуированном графе / Урал. гос. ун-т., 1994. 27с. Деп. в ВИНИТИ. 21.10.94. №2391-В94.

14. Щербакова В.А. Алгоритм решения мощностной задачи Штейнера на ориентированном градуированном графе методом динамического программирования / Урал. гос. ун-т., 1995. 12с. Деп. в ВИНИТИ. 14.04.95. №1052-В95.

15. Ахо Х., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.

16. Щербакова В.А. Классы ориентированных градуированных графов с полиномиально разрешимой мощностной задачей Штейнера // Дискрет. математика. 1997. Т.9, №4. С.73–85.

Статья поступила 01.12.1997 г.